Goal of the project:

Build a forecasting model to predict weekly or monthly sales.

Output:

1. Clean EDA (seasonality, trends)
2. Forecasting plots (next 12–24 weeks)
3. Model comparison table (RMSE, MAE)
4. Insights like:

“Holiday weeks show 20% increase in sales.”

“Store 2 has a clear annual peak in November.”

Data Importing

In [1]:
```
import numpy as np
import pandas as pd

dataset=pd.read_csv('Documents/My_projetcs/project_3/Walmart.csv')
dataset.head()
```

Out[1]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemplo
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	

In [2]:
```
dataset1=dataset.copy()
dataset1.head()
```

Out[2]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemplo
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	

Data Cleaning

In [3]:
```
dataset1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Store         6435 non-null   int64  
 1   Date          6435 non-null   object 
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64  
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

In [4]:
```
dataset1['Date']=pd.to_datetime(dataset1['Date'],format='%d-%m-%Y')
```

In [5]:
```
dataset1['Date'] = pd.to_datetime(dataset1['Date'])
dataset1['Year'] = dataset1['Date'].dt.year
dataset1['Month'] = dataset1['Date'].dt.month
dataset1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   datetime64[ns]
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
 8   Year          6435 non-null   int32
 9   Month         6435 non-null   int32
dtypes: datetime64[ns](1), float64(5), int32(2), int64(2)
memory usage: 452.6 KB
```

In [6]:
```
dataset1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   datetime64[ns]
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
 8   Year          6435 non-null   int32
 9   Month         6435 non-null   int32
dtypes: datetime64[ns](1), float64(5), int32(2), int64(2)
memory usage: 452.6 KB
```

In [7]:
```
dataset1.head()
```

Out[7]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unempl
0	1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	
1	1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	
4	1	2010-03-05	1554806.68	0	46.50	2.625	211.350143	

In [8]:
```
dataset1.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
dataset1.isnull().sum()
```

Out[9]:
```
Store           0
Date            0
Weekly_Sales    0
Holiday_Flag    0
Temperature     0
Fuel_Price      0
CPI             0
Unemployment    0
Year            0
Month           0
dtype: int64
```

In [10]:
```
dataset1.nunique()
```

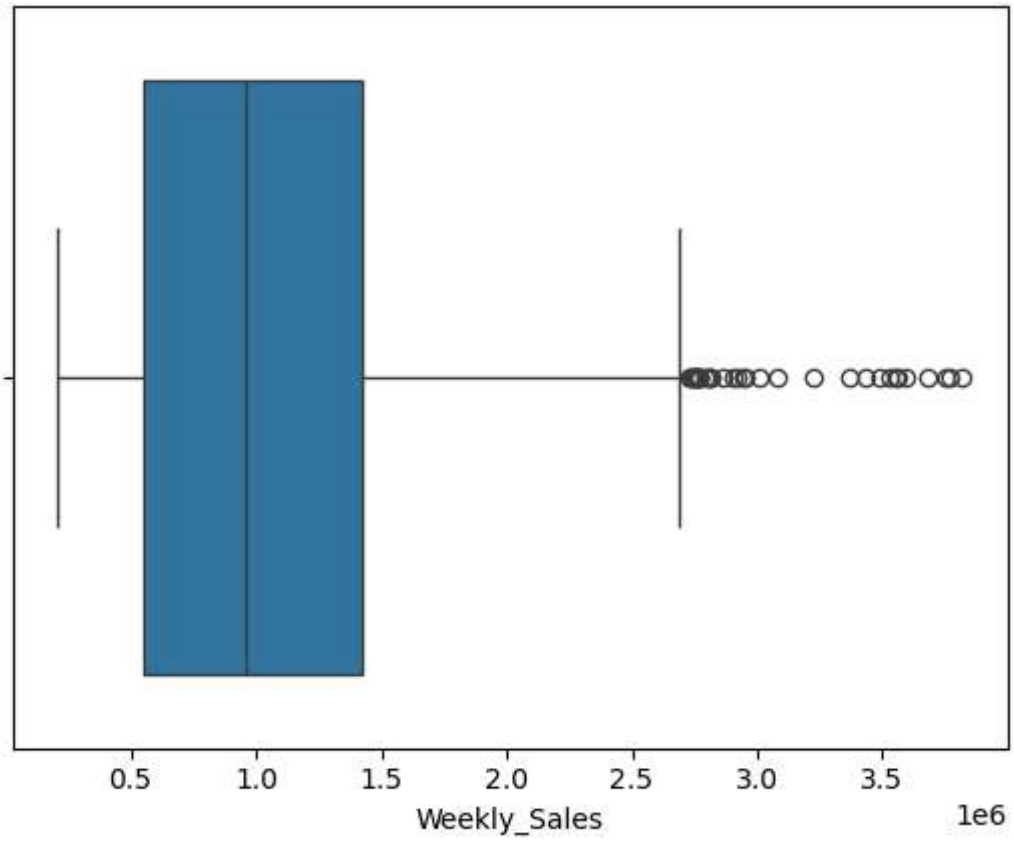
Out[10]:
```
Store            45
Date            143
Weekly_Sales   6435
Holiday_Flag      2
Temperature    3528
Fuel_Price      892
CPI            2145
Unemployment    349
Year              3
Month            12
dtype: int64
```

In [11]:
```
dataset1.describe()
```

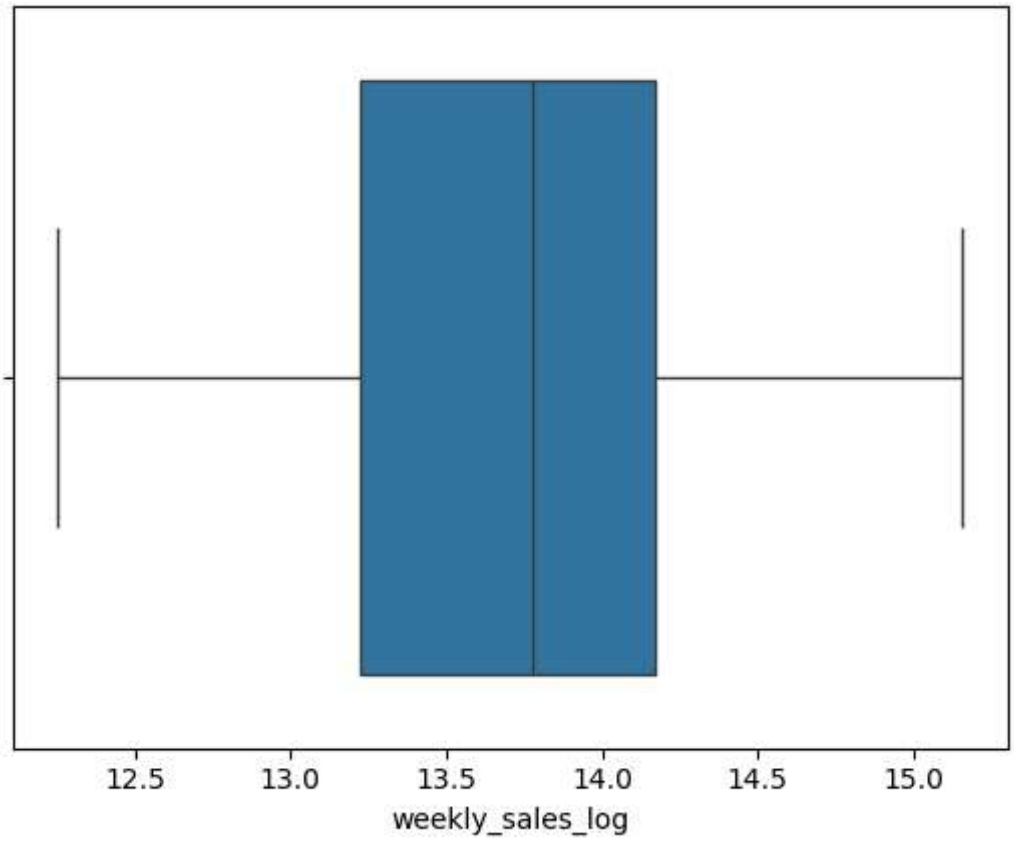
Out[11]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	
count	6435.000000	6435	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.0
mean	23.000000	2011-06-17 00:00:00	1.046965e+06	0.069930	60.663782	3.358607	171.5
min	1.000000	2010-02-05 00:00:00	2.099862e+05	0.000000	-2.060000	2.472000	126.0
25%	12.000000	2010-10-08 00:00:00	5.533501e+05	0.000000	47.460000	2.933000	131.7
50%	23.000000	2011-06-17 00:00:00	9.607460e+05	0.000000	62.670000	3.445000	182.6
75%	34.000000	2012-02-24 00:00:00	1.420159e+06	0.000000	74.940000	3.735000	212.7
max	45.000000	2012-10-26 00:00:00	3.818686e+06	1.000000	100.140000	4.468000	227.2
std	12.988182	NaN	5.643666e+05	0.255049	18.444933	0.459020	39.3

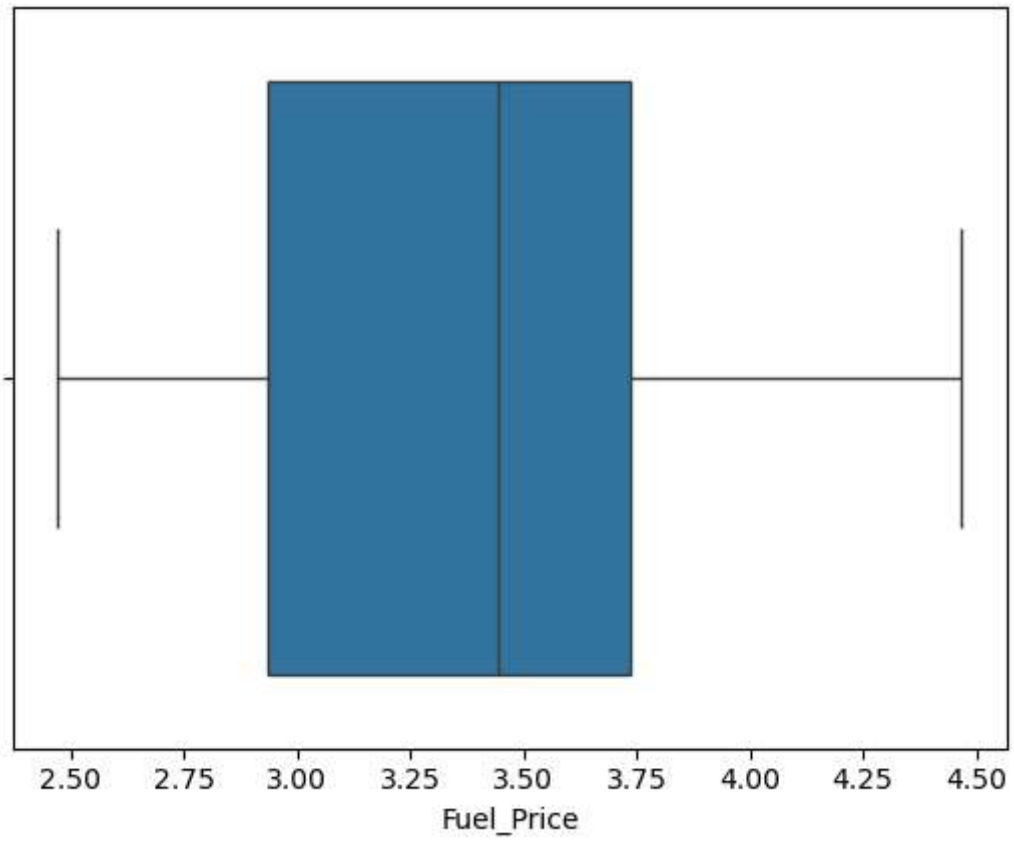
In [12]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.boxplot(data=dataset1,x='Weekly_Sales')
plt.show()
```



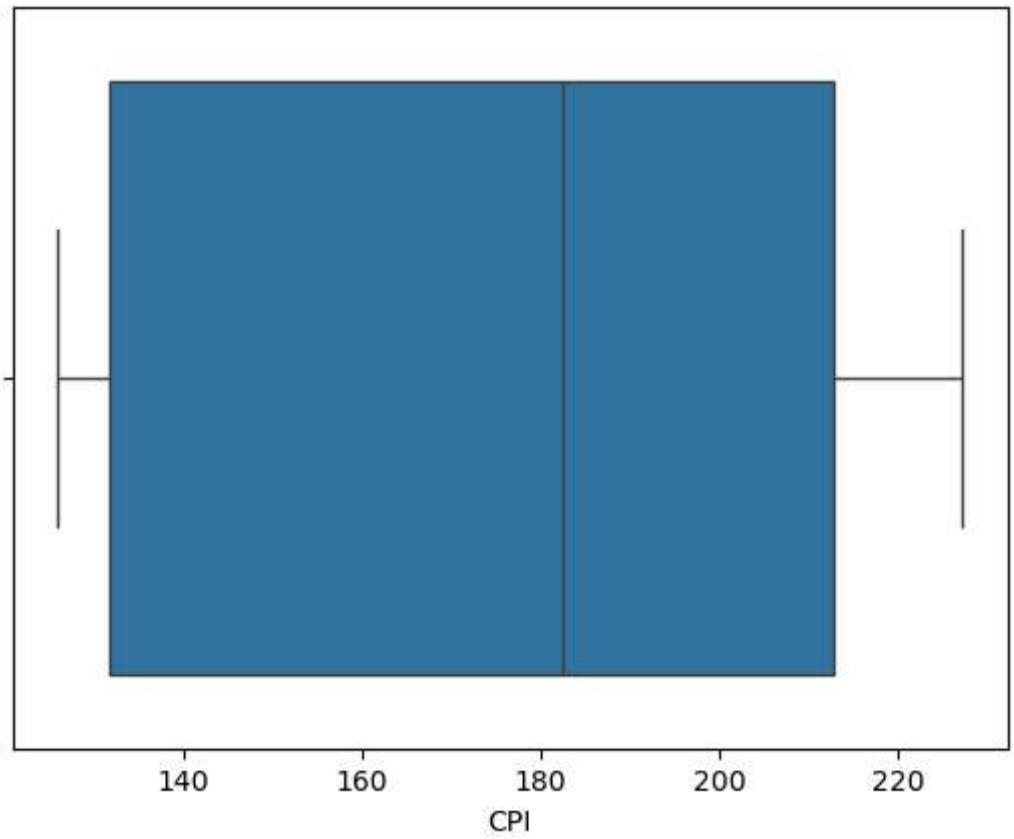
In [13]:
```
# log transformation
dataset1['weekly_sales_log']=np.log1p(dataset1['Weekly_Sales'])
sns.boxplot(data=dataset1,x='weekly_sales_log')
plt.show()
```



In [14]:
```
sns.boxplot(data=dataset1,x='Fuel_Price')
plt.show()
```



In [15]:
```
sns.boxplot(data=dataset1,x='CPI')
plt.show()
```



EDA

In [16]:
```
dataset1.head()
```

Out[16]:

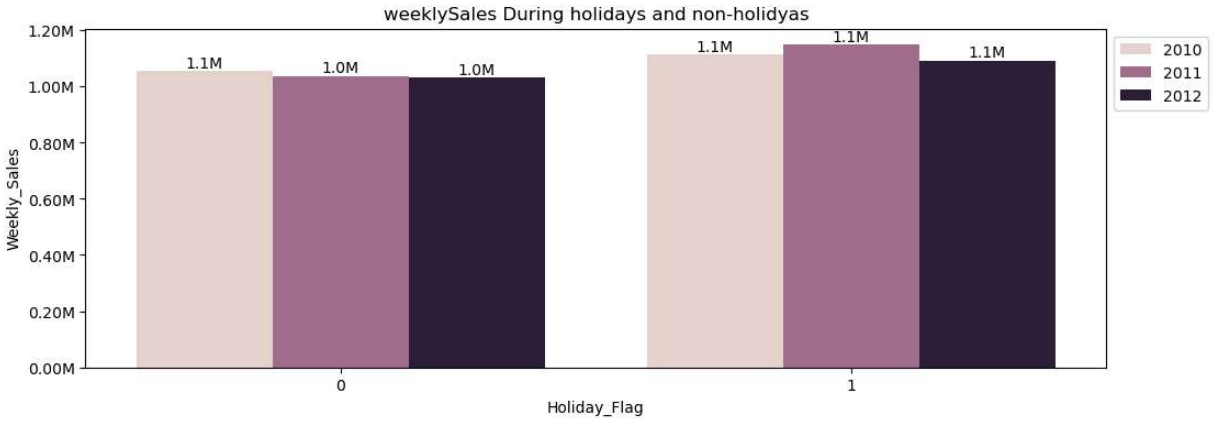
	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemplo
0	1	2010-02-05	1643690.90	0	42.31	2.572	211.096358	
1	1	2010-02-12	1641957.44	1	38.51	2.548	211.242170	
2	1	2010-02-19	1611968.17	0	39.93	2.514	211.289143	
3	1	2010-02-26	1409727.59	0	46.63	2.561	211.319643	
4	1	2010-03-05	1554806.68	0	46.50	2.625	211.350143	

In [17]:
```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

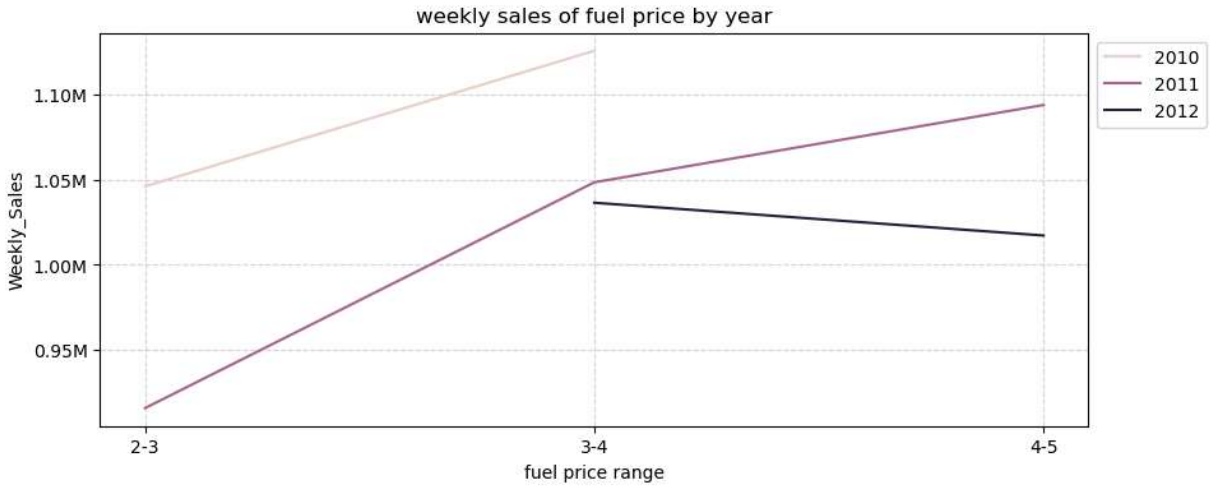
```
from matplotlib.ticker import FuncFormatter

def yaxis_formatter(x,pos):
    return f'{int(x/1000000)}M'


storesales=dataset1.groupby(['Store','Year'])['Weekly_Sales'].sum().reset_index()

plt.figure(figsize=(15,4))
sns.lineplot(data=storesales,x='Store',y='Weekly_Sales',hue='Year',style='Year',das
plt.gca().yaxis.set_major_formatter(FuncFormatter(yaxis_formatter))
plt.xticks(range(1,46))
plt.margins(x=0)
plt.grid(True,linestyle='--',alpha=0.7)
plt.title('Weekly sales of stores')
plt.show()
```

In [18]:

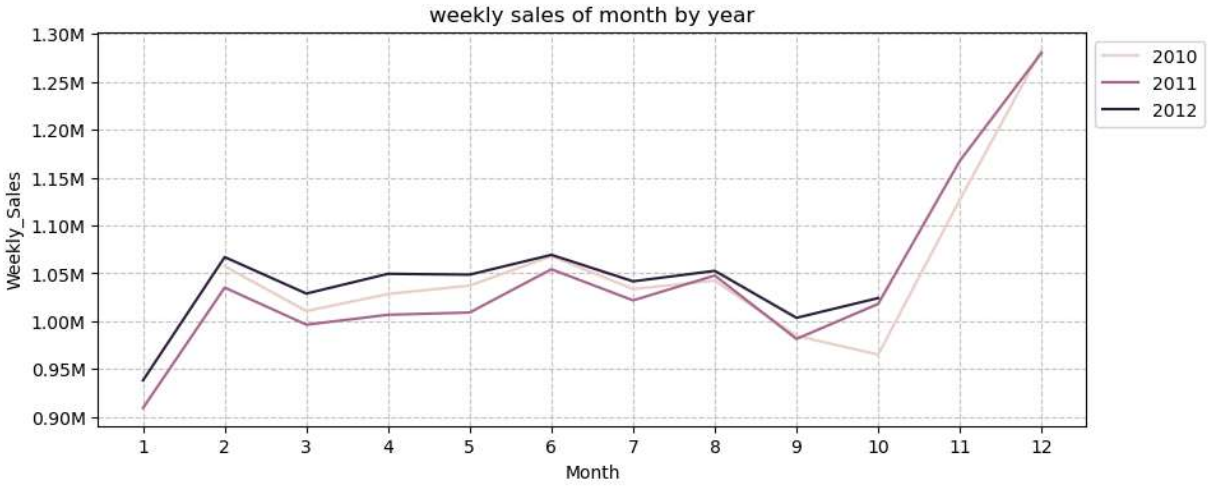
```
def ylabels(x,pos):
    return f'{x/1000000:.2f}M'



plt.figure(figsize=(12,4))
ax=sns.barplot(data=dataset1,x='Holiday_Flag',y='Weekly_Sales',hue='Year',errorbar=

for container in ax.containers:
    ax.bar_label(container,labels=[f'{v.get_height()/1000000:.1f}M' for v in contai



plt.title('weeklySales During holidays and non-holidyas')
plt.gca().yaxis.set_major_formatter(FuncFormatter(ylabels))
plt.legend(bbox_to_anchor=(1,1),loc='upper left')
plt.show()
```



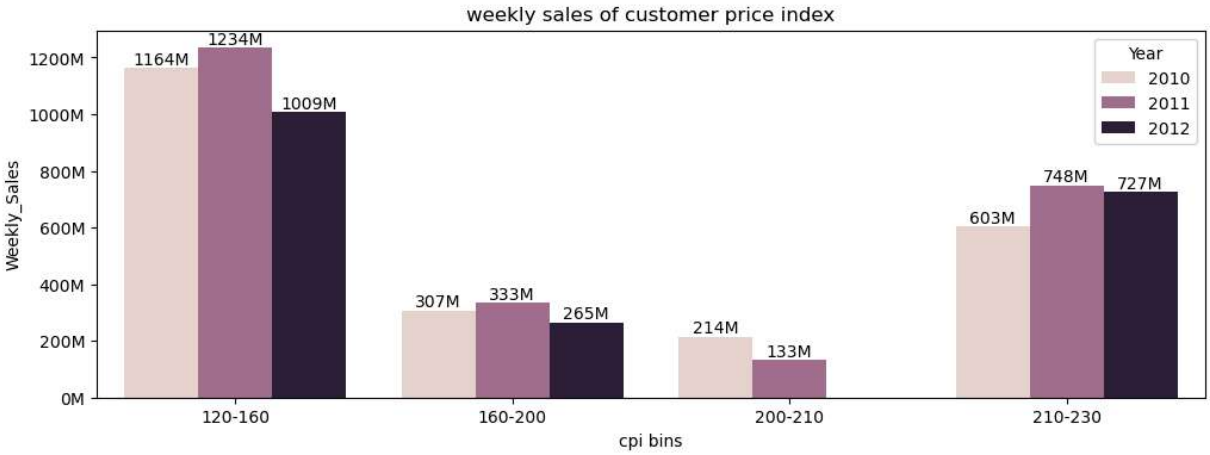
In [19]:
```
bins=[2,3,4,5]

df=dataset1.copy()
df['Fuel_bins']=pd.cut(df['Fuel_Price'],bins=bins,labels=['2-3','3-4','4-5'])

plt.figure(figsize=(10,4))
ax=sns.lineplot(data=df,x='Fuel_bins',y='Weekly_Sales',hue='Year',errorbar=None)
plt.gca().yaxis.set_major_formatter(FuncFormatter(ylabels))
plt.title('weekly sales of fuel price by year')
plt.xlabel('fuel price range')
plt.legend(bbox_to_anchor=(1,1),loc='upper left')
plt.grid(True,linestyle='--',alpha=0.5)
plt.show()
```



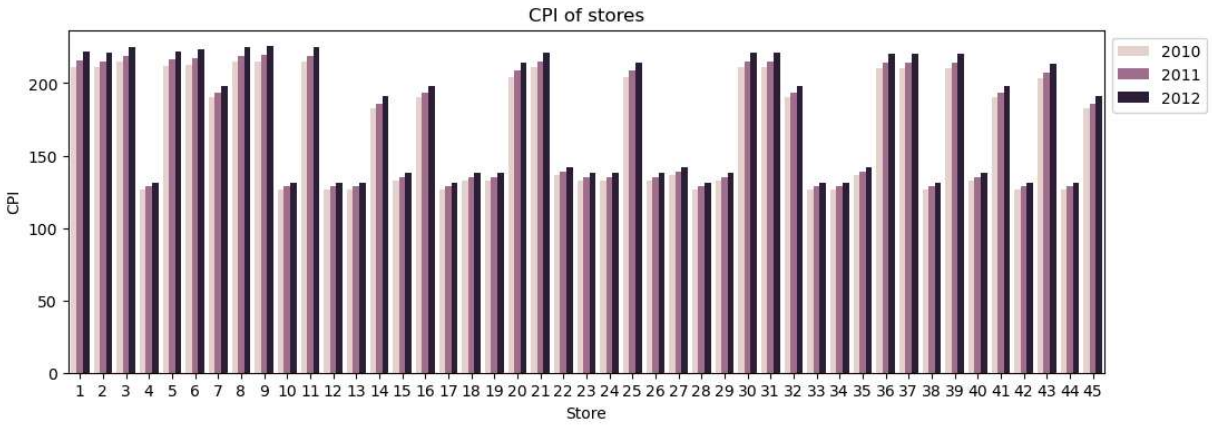
In [20]:
```
plt.figure(figsize=(10,4))
sns.lineplot(data=dataset1,x='Month',y='Weekly_Sales',hue='Year',errorbar=None)
plt.title('weekly sales of month by year')
plt.gca().yaxis.set_major_formatter(FuncFormatter(ylabels))
plt.xticks(range(1,13))
plt.legend(bbox_to_anchor=(1,1),loc='upper left')
plt.grid(True,linestyle='--',alpha=0.7)
plt.show()
```



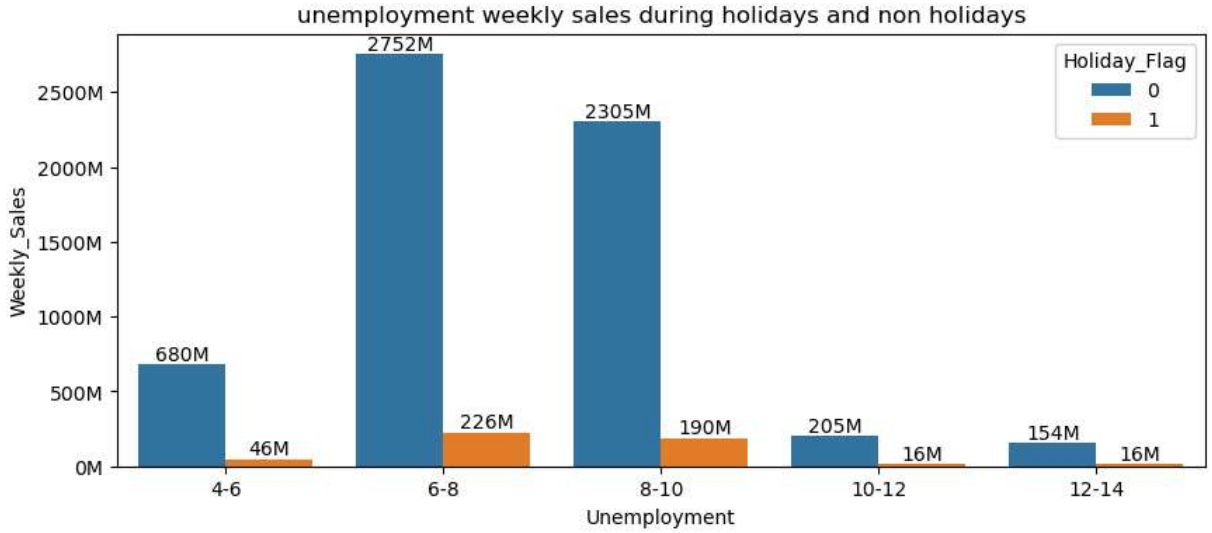
In [21]:
```
bins=[120,160,200,210,230]
df['cpi bins']=pd.cut(
    df['CPI'],bins=bins,labels=['120-160','160-200','200-210','210-230'])

plt.figure(figsize=(12,4))
ax=sns.barplot(data=df,x='cpi bins',y='Weekly_Sales',hue='Year',errorbar=None,estim

for container in ax.containers:
    ax.bar_label(container,labels=[f'{v.get_height()/1000000:.0f}M' for v in contai

plt.title('weekly sales of customer price index')
plt.gca().yaxis.set_major_formatter(FuncFormatter(yaxis_formatter))
plt.show()
```



In [22]:
```
plt.figure(figsize=(12,4))
sns.barplot(data=dataset1,x='Store',y='CPI',hue='Year',errorbar=None)
plt.title('CPI of stores')
plt.legend(bbox_to_anchor=(1,1),loc='upper left')
plt.show()
```



In [23]:
```
bins=[4,6,8,10,12,14]
df['unemployment bins']=pd.cut(df['Unemployment'],bins=bins,labels=['4-6','6-8','8-


plt.figure(figsize=(10,4))
ax=sns.barplot(data=df,x='unemployment bins',y='Weekly_Sales',hue='Holiday_Flag',er

for container in ax.containers:
    ax.bar_label(container,labels=[f'{v.get_height()/1000000:.0f}M' for v in contai

plt.title('unemployment weekly sales during holidays and non holidays')
plt.gca().yaxis.set_major_formatter(FuncFormatter(yaxis_formatter))
plt.xlabel('Unemployment')
plt.show()
```



Feature Engineering

In [24]:
```
dataset1['Date'] = pd.to_datetime(df['Date'])
dataset1['Week'] = dataset1['Date'].dt.isocalendar().week
dataset1['Day'] = dataset1['Date'].dt.day
dataset1['DayOfWeek'] = dataset1['Date'].dt.dayofweek
dataset1=dataset1.drop(['Date'],axis=1)
dataset1.head()
```

Out[24]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	1554806.68	0	46.50	2.625	211.350143	8.106

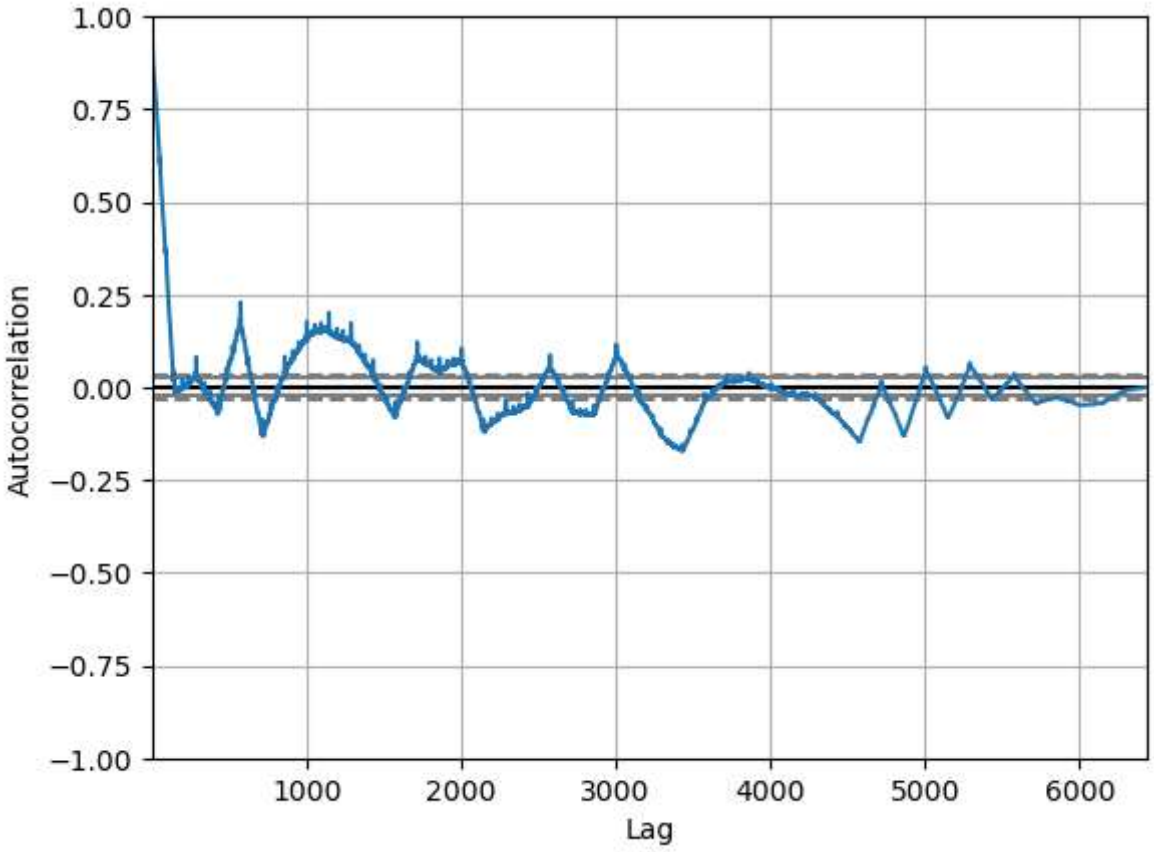
Forecasting Models

ADF test

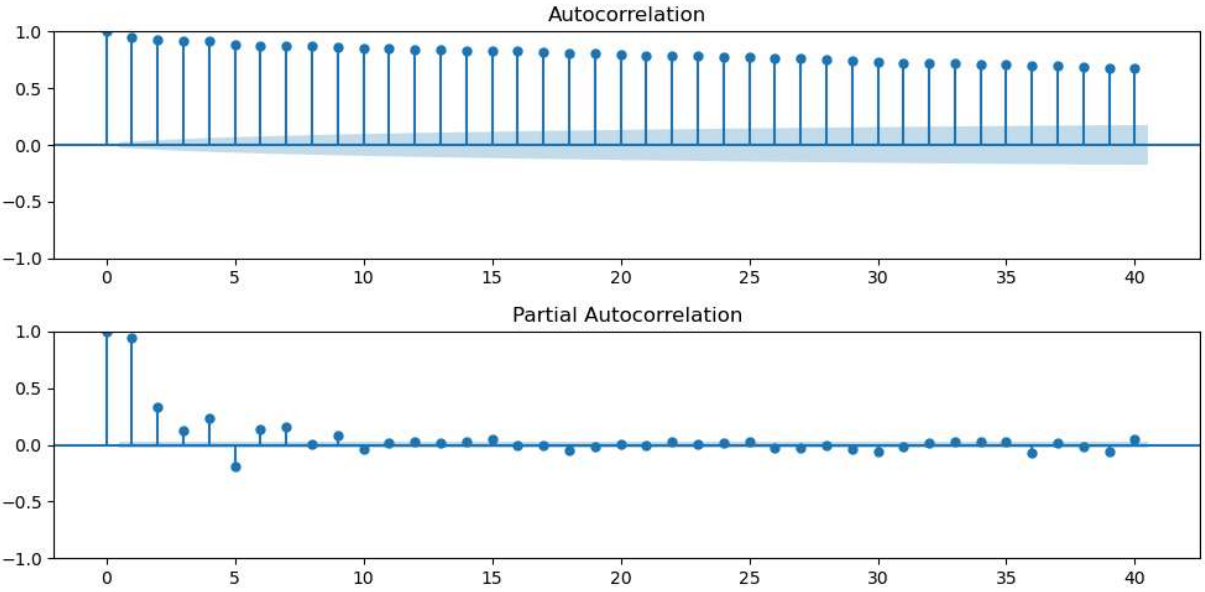
In [25]:
```
from statsmodels.tsa.stattools import adfuller

test_result=adfuller(dataset1['Weekly_Sales'])

def adfuller_test(Weekly_Sales):
    result=adfuller(Weekly_Sales)
    labels=['Adf test statistics','p_values','Lags used','no.of observations used']
    for values,label in zip(result,labels):
        print(label+' : '+str(values))
    if result[1] <= 0.05:
        print('stationary')
    else:
        print('not stationary')

adfuller_test(dataset1['Weekly_Sales'])
```

```
Adf test statistics : -4.624149498578308
p_values : 0.00011655869699901031
Lags used : 34
no.of observations used : 6400
stationary
```

In [26]:
```
from pandas.plotting import autocorrelation_plot
autocorrelation_plot(dataset1['Weekly_Sales'])
plt.show()
```



In [27]:
```
from statsmodels.graphics.tsaplots import plot_acf,plot_pacf

fig,ax=plt.subplots(2,1,figsize=(10,5))
plot_acf(dataset1['Weekly_Sales'],lags=40,ax=ax[0])
plot_pacf(dataset1['Weekly_Sales'],lags=40,ax=ax[1])
plt.subplots_adjust(hspace=0.5)
plt.tight_layout()
plt.show()
```



In [28]:
```
dataset1['Weekly_Sales_scaled']=dataset1['Weekly_Sales']/1e6
dataset1.head()
```

Out[28]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	1554806.68	0	46.50	2.625	211.350143	8.106

ARIMA MODEL

In [29]:
```
from statsmodels.tsa.arima.model import ARIMA

arima_model=ARIMA(dataset1['Weekly_Sales_scaled'],order=(1,0,1))
arima_model_fit=arima_model.fit()
```

In [30]:
```
arima_model_fit.summary()
```

Out[30]:

SARIMAX Results

Dep. Variable:	Weekly_Sales_scaled	**No. Observations:**	6435
Model:	ARIMA(1, 0, 1)	**Log Likelihood**	2240.859
Date:	Tue, 14 Oct 2025	**AIC**	-4473.717
Time:	18:58:52	**BIC**	-4446.639
Sample:	0	**HQIC**	-4464.347
	- 6435		
Covariance Type:	opg		

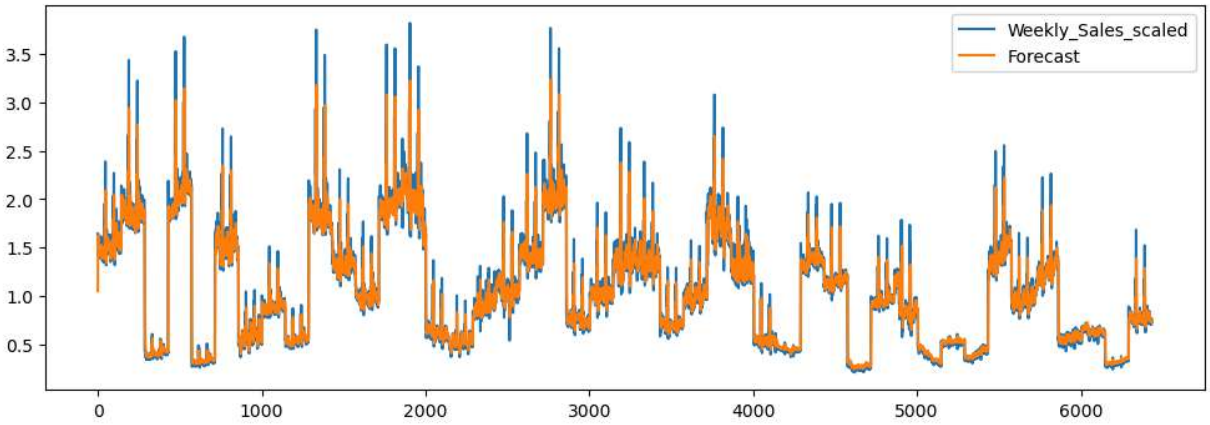
	coef	std err	z	P>|z|	[0.025	0.975]
const	1.0497	0.114	9.195	0.000	0.826	1.273
ar.L1	0.9856	0.002	479.521	0.000	0.982	0.990
ma.L1	-0.4547	0.005	-90.581	0.000	-0.465	-0.445
sigma2	0.0292	0.000	170.430	0.000	0.029	0.030

Ljung-Box (L1) (Q):	4.62	**Jarque-Bera (JB):**	120859.40
Prob(Q):	0.03	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.36	**Skew:**	0.07
Prob(H) (two-sided):	0.00	**Kurtosis:**	24.23

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [31]:
```
dataset1['Forecast']=arima_model_fit.predict(start=0,end=len(dataset1)-1,dynamic=Fa
dataset1['Forecast_dynamic'] = arima_model_fit.predict(start=0, end=len(dataset1)-1
dataset1[['Weekly_Sales_scaled','Forecast']].plot(figsize=(12,4))
```

Out[31]: <Axes: >



In [32]:
```
forecast_12weeks_arima=arima_model_fit.forecast(steps=12)
forecast_12weeks_rescaled_arima=forecast_12weeks_arima*1e6

forecast_24weeks_arima=arima_model_fit.forecast(steps=24)
forecast_24weeks_rescaled_arima=forecast_24weeks_arima*1e6

forecast_12weeks_rescaled_arima, forecast_24weeks_rescaled_arima
```

Out[32]:
```
(6435    751783.813804
 6436    756061.869274
 6437    760278.493214
 6438    764434.567762
 6439    768530.962389
 6440    772568.534078
 6441    776548.127510
 6442    780470.575233
 6443    784336.697842
 6444    788147.304149
 6445    791903.191348
 6446    795605.145190
 Name: predicted_mean, dtype: float64,
 6435    751783.813804
 6436    756061.869274
 6437    760278.493214
 6438    764434.567762
 6439    768530.962389
 6440    772568.534078
 6441    776548.127510
 6442    780470.575233
 6443    784336.697842
 6444    788147.304149
 6445    791903.191348
 6446    795605.145190
 6447    799253.940141
 6448    802850.339545
 6449    806395.095787
 6450    809888.950445
 6451    813332.634451
 6452    816726.868239
 6453    820072.361898
 6454    823369.815322
 6455    826619.918354
 6456    829823.350929
 6457    832980.783221
 6458    836092.875780
 Name: predicted_mean, dtype: float64)
```

In [33]:
```
forecast_12_to_24_arima=forecast_24weeks_rescaled_arima[12:24]
future_12_to_24weeks_arima=range(len(dataset1)+12,len(dataset1)+24)

forecast_12_to_24df_arima=pd.DataFrame({
    'Week':future_12_to_24weeks_arima,
    'Forecast':forecast_12_to_24_arima.values
})
print(forecast_12_to_24df_arima)
```

```
    Week       Forecast
0   6447  799253.940141
1   6448  802850.339545
2   6449  806395.095787
3   6450  809888.950445
4   6451  813332.634451
5   6452  816726.868239
6   6453  820072.361898
7   6454  823369.815322
8   6455  826619.918354
9   6456  829823.350929
10  6457  832980.783221
11  6458  836092.875780
```

In [34]:
```
future_weeks_12_arima = range(len(dataset1), len(dataset1) + 12)
forecast_df_12_arima = pd.DataFrame({
    'Week': future_weeks_12_arima,
    'Forecast': forecast_12weeks_rescaled_arima.values
})

future_weeks_24_arima = range(len(dataset1), len(dataset1) + 24)
forecast_df_24_arima = pd.DataFrame({
    'Week': future_weeks_24_arima,
    'Forecast': forecast_24weeks_rescaled_arima.values
})
```

In [35]:
```
forecast_df_12_arima
```

Out[35]:

	Week	Forecast
0	6435	751783.813804
1	6436	756061.869274
2	6437	760278.493214
3	6438	764434.567762
4	6439	768530.962389
5	6440	772568.534078
6	6441	776548.127510
7	6442	780470.575233
8	6443	784336.697842
9	6444	788147.304149
10	6445	791903.191348
11	6446	795605.145190

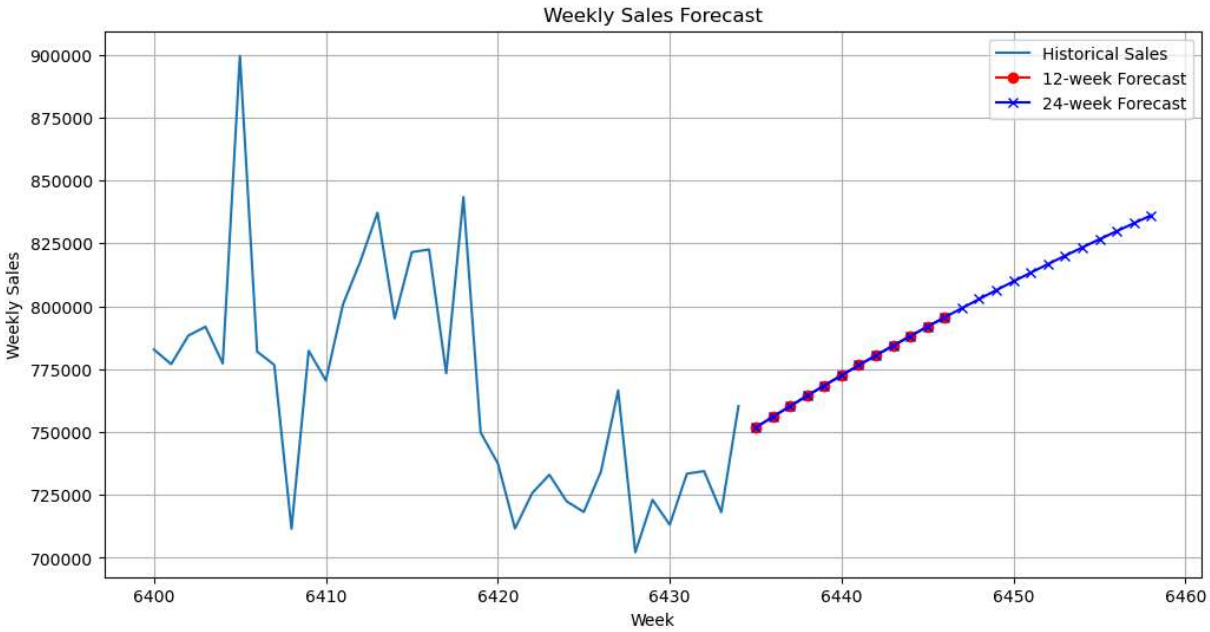
In [36]:
```
forecast_df_24_arima
```

Out[36]:

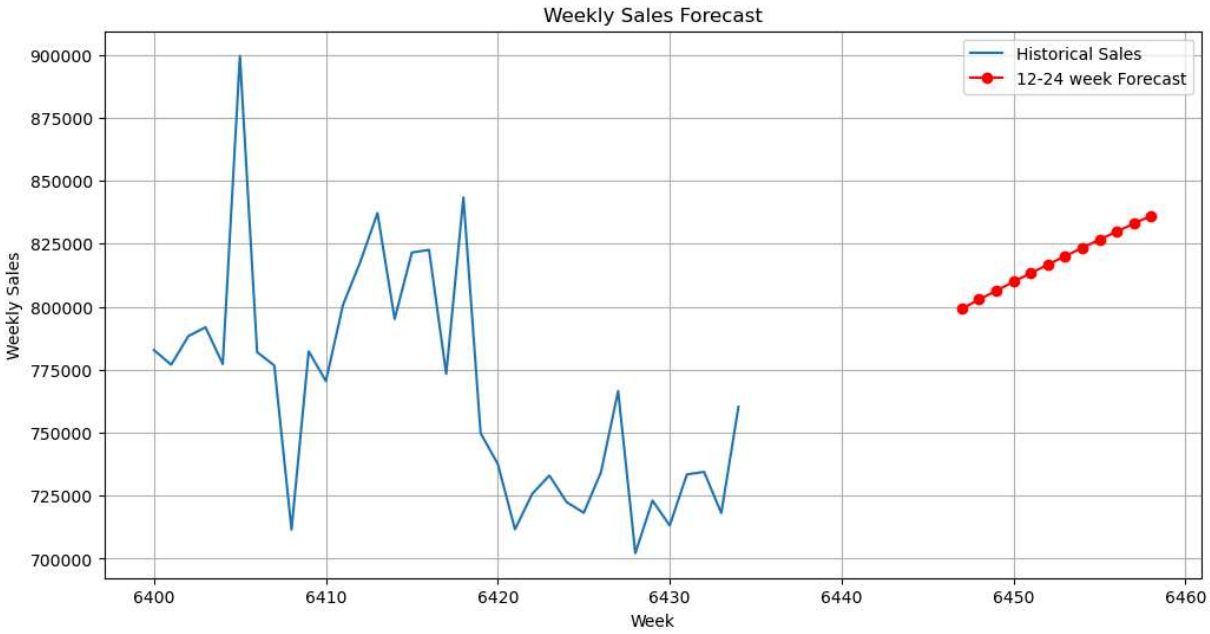
	Week	Forecast
0	6435	751783.813804
1	6436	756061.869274
2	6437	760278.493214
3	6438	764434.567762
4	6439	768530.962389
5	6440	772568.534078
6	6441	776548.127510
7	6442	780470.575233
8	6443	784336.697842
9	6444	788147.304149
10	6445	791903.191348
11	6446	795605.145190
12	6447	799253.940141
13	6448	802850.339545
14	6449	806395.095787
15	6450	809888.950445
16	6451	813332.634451
17	6452	816726.868239
18	6453	820072.361898
19	6454	823369.815322
20	6455	826619.918354
21	6456	829823.350929
22	6457	832980.783221
23	6458	836092.875780

In [37]:
```
start_index = 6400

plt.figure(figsize=(12,6))

plt.plot(range(start_index,len(dataset1)), dataset1['Weekly_Sales'][start_index:],

# Plot 12-week forecast
plt.plot(forecast_df_12_arima['Week'], forecast_df_12_arima['Forecast'], label='12-

# Plot 24-week forecast
```

```
plt.plot(forecast_df_24_arima['Week'], forecast_df_24_arima['Forecast'], label='24-

plt.xlabel('Week')
plt.ylabel('Weekly Sales')
plt.title('Weekly Sales Forecast')
plt.legend()
plt.grid(True)
plt.show()
```



In [38]:
```
start_index = 6400

plt.figure(figsize=(12,6))

plt.plot(range(start_index,len(dataset1)), dataset1['Weekly_Sales'][start_index:],

# Plot 12-week forecast
plt.plot(forecast_12_to_24df_arima['Week'], forecast_12_to_24df_arima['Forecast'],
plt.xlabel('Week')
plt.ylabel('Weekly Sales')
plt.title('Weekly Sales Forecast')
plt.legend()
plt.grid(True)
plt.show()
```



SARIMA MODEL

In [39]:
```
from statsmodels.tsa.statespace.sarimax import SARIMAX

sarima_model=SARIMAX(
    dataset1['Weekly_Sales_scaled'],
    order=(1,0,1),
    seasonal_order=(1,1,1,52),
    enforce_stationarity=False,
    enforce_invertibility=False)
results=sarima_model.fit()
```

In [40]:
```
results.summary()
```

Out[40]:

SARIMAX Results

Dep. Variable:	Weekly_Sales_scaled	**No. Observations:**	6435
Model:	SARIMAX(1, 0, 1)x(1, 1, 1, 52)	**Log Likelihood**	2641.589
Date:	Tue, 14 Oct 2025	**AIC**	-5273.177
Time:	19:02:28	**BIC**	-5239.413
Sample:	0	**HQIC**	-5261.483
	- 6435		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9806	0.002	488.882	0.000	0.977	0.985
ma.L1	-0.4205	0.006	-76.422	0.000	-0.431	-0.410
ar.S.L52	0.3063	0.006	50.952	0.000	0.294	0.318
ma.S.L52	-1.0138	0.006	-158.953	0.000	-1.026	-1.001
sigma2	0.0240	0.000	102.651	0.000	0.024	0.024

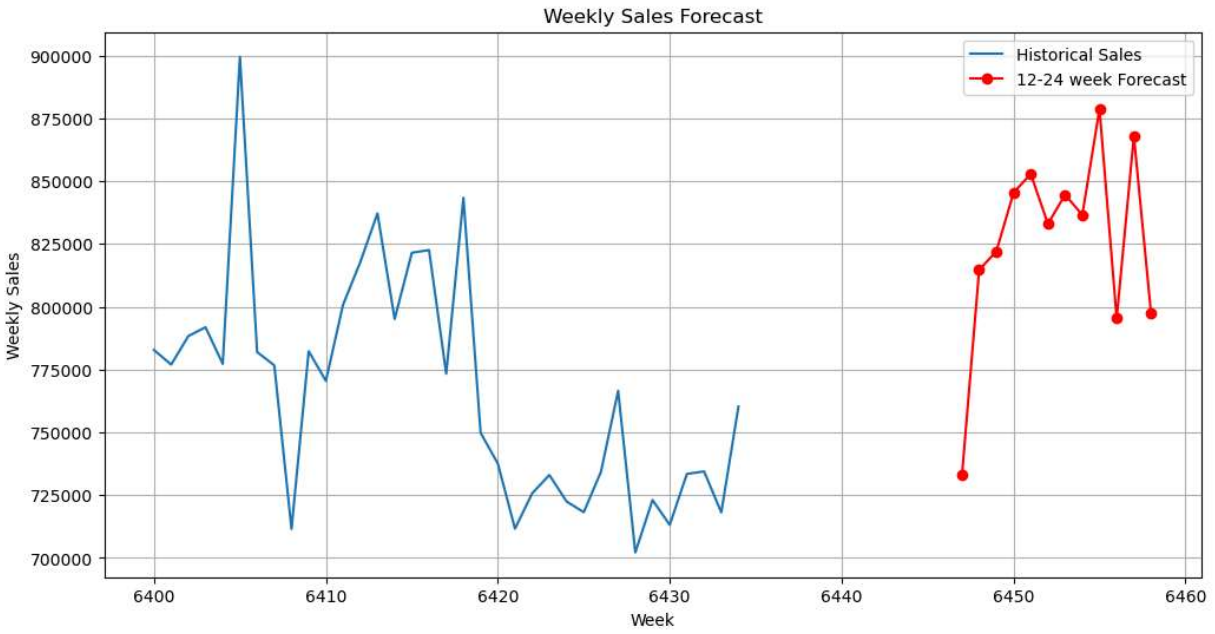
Ljung-Box (L1) (Q):	4.02	**Jarque-Bera (JB):**	120289.67
Prob(Q):	0.05	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.36	**Skew:**	-0.20
Prob(H) (two-sided):	0.00	**Kurtosis:**	24.35

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [42]:
```
forecast_24weeks_sarima = results.forecast(steps=24)
forecast_24weeks_original_sarima= forecast_24weeks_sarima * 1e6
forecast_12_to_24_sarima= forecast_24weeks_original_sarima[12:24]
future_weeks_12_to_24_sarima= range(len(dataset1) + 12, len(dataset1) + 24)


forecast_df_12_to_24_sarima= pd.DataFrame({
    'Week': future_weeks_12_to_24_sarima,
    'Forecast': forecast_12_to_24_sarima.values
})

print(forecast_df_12_to_24_sarima)
```

	Week	Forecast
0	6447	732876.238420
1	6448	814669.106775
2	6449	822062.402001
3	6450	845381.706430
4	6451	852892.487905
5	6452	832982.161536
6	6453	844616.734160
7	6454	836540.151907
8	6455	878748.019585
9	6456	795366.174188
10	6457	867907.527130
11	6458	797306.452957

In [43]:
```
start_index = 6400

plt.figure(figsize=(12,6))

plt.plot(range(start_index,len(dataset1)), dataset1['Weekly_Sales'][start_index:],

# Plot 12-week forecast
plt.plot(forecast_df_12_to_24_sarima['Week'], forecast_df_12_to_24_sarima['Forecast
plt.xlabel('Week')
plt.ylabel('Weekly Sales')
plt.title('Weekly Sales Forecast')
plt.legend()
plt.grid(True)
plt.show()
```



PROPHET model

In [44]:
```
from prophet import Prophet

df_prophet = df.reset_index()
```

```
df_prophet.rename(columns={'Date':'ds','Weekly_Sales':'y'},inplace=True)
df_prophet['ds'] = pd.to_datetime(df_prophet['ds'])
```

In [45]:
```
prophet_model = Prophet(
    yearly_seasonality=True,
    weekly_seasonality=False,
    daily_seasonality=False
)


prophet_model.fit(df_prophet)
```

```
19:03:16 - cmdstanpy - INFO - Chain [1] start processing
19:03:17 - cmdstanpy - INFO - Chain [1] done processing
```

Out[45]:
```
<prophet.forecaster.Prophet at 0x2af4a2ecfd0>
```

In [61]:
```
future = prophet_model.make_future_dataframe(periods=24, freq='W-FRI')

forecast = prophet_model.predict(future)
forecast
```

Out[61]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	addit
0	2010-02-05	1.054202e+06	363246.141939	1.775647e+06	1.054202e+06	1.054202e+06	149
1	2010-02-12	1.053887e+06	382856.762356	1.732407e+06	1.053887e+06	1.053887e+06	298
2	2010-02-19	1.053573e+06	310333.091091	1.733439e+06	1.053573e+06	1.053573e+06	-9
3	2010-02-26	1.053258e+06	280036.382232	1.740087e+06	1.053258e+06	1.053258e+06	-275
4	2010-03-05	1.052943e+06	293226.185704	1.757887e+06	1.052943e+06	1.052943e+06	-334
...	...	...	...	...	...	...	
162	2013-03-15	1.082215e+06	364124.807059	1.738763e+06	1.081237e+06	1.083218e+06	-377
163	2013-03-22	1.082584e+06	318117.339674	1.731380e+06	1.081506e+06	1.083672e+06	-454
164	2013-03-29	1.082953e+06	313606.881229	1.753933e+06	1.081783e+06	1.084096e+06	-401
165	2013-04-05	1.083322e+06	376708.681443	1.754075e+06	1.082070e+06	1.084580e+06	-188
166	2013-04-12	1.083692e+06	332267.362478	1.831913e+06	1.082364e+06	1.085033e+06	-35

167 rows × 16 columns

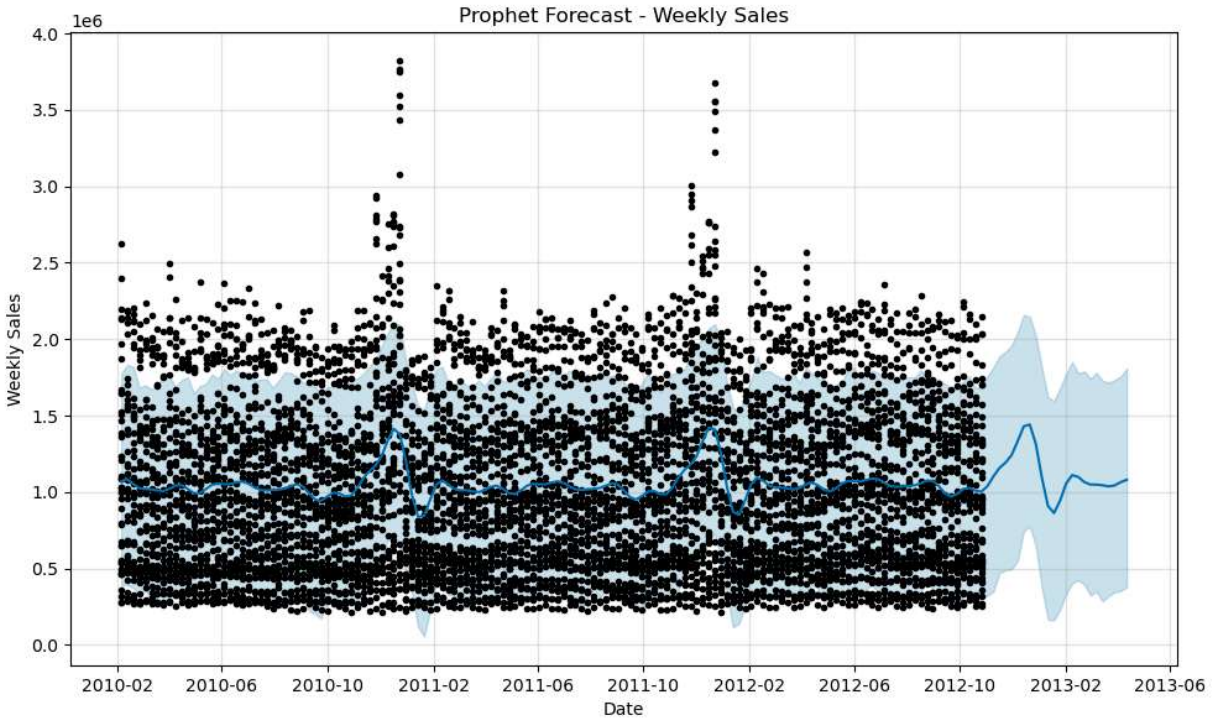
In [47]:
```
forecast_future = forecast[forecast['ds'] > df_prophet['ds'].max()]
```

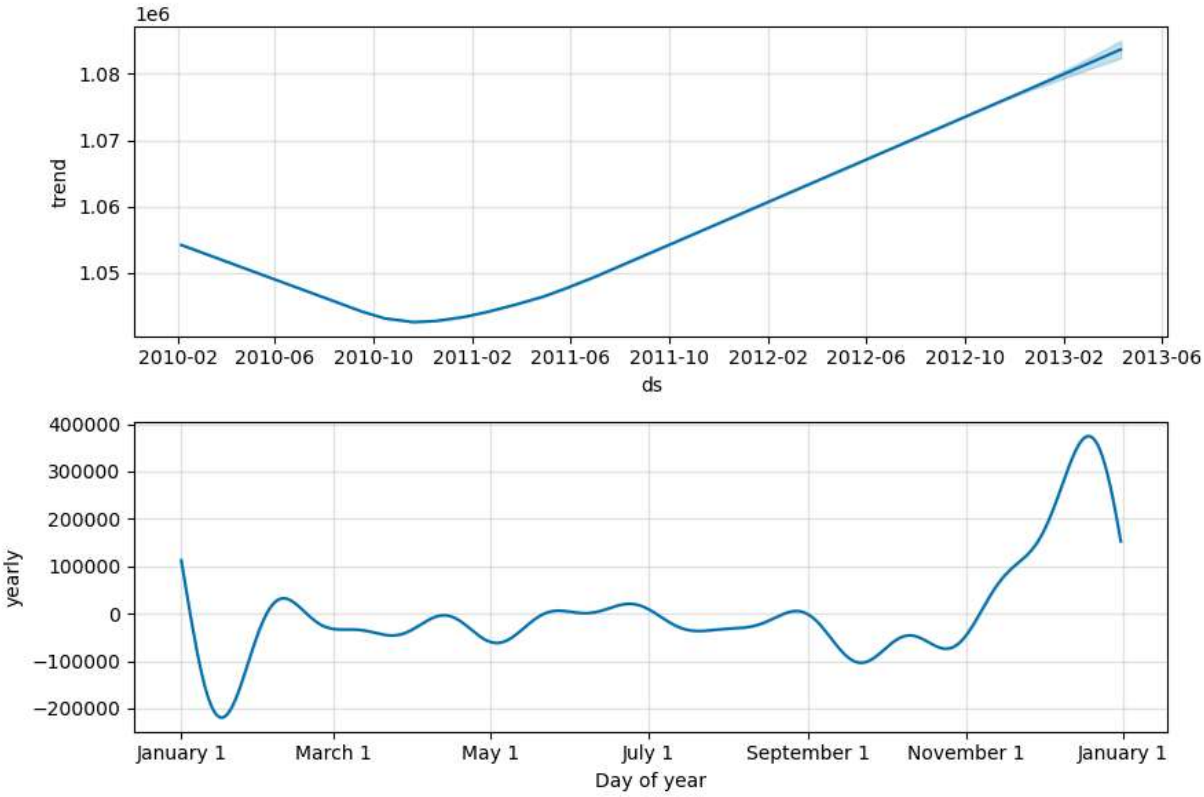
In [48]:
```
forecast_12_to_24_prophet= forecast_future.iloc[12:24].copy()
forecast_12_to_24_prophet= forecast_12_to_24_prophet[['ds', 'yhat', 'yhat_lower', '
forecast_12_to_24_prophet.rename(columns={'ds': 'Week', 'yhat': 'Forecast'}, inplac
```

In [49]:
```
print(forecast_12_to_24_prophet)
```

	Week	Forecast	yhat_lower	yhat_upper
155	2013-01-25	9.438506e+05	231827.818888	1.680279e+06
156	2013-02-01	1.055792e+06	334974.508696	1.773687e+06
157	2013-02-08	1.111040e+06	403842.525057	1.849625e+06
158	2013-02-15	1.098663e+06	422479.714122	1.777089e+06
159	2013-02-22	1.065186e+06	397001.914384	1.788751e+06
160	2013-03-01	1.049198e+06	324476.009131	1.742640e+06
161	2013-03-08	1.048526e+06	349397.385822	1.780573e+06
162	2013-03-15	1.044489e+06	285931.831865	1.728402e+06
163	2013-03-22	1.037094e+06	320629.056263	1.716120e+06
164	2013-03-29	1.042822e+06	345101.980083	1.733005e+06
165	2013-04-05	1.064493e+06	353487.121040	1.756515e+06
166	2013-04-12	1.080111e+06	377536.697480	1.809363e+06

In [50]:
```
prophet_model.plot(forecast)
plt.title('Prophet Forecast - Weekly Sales')
plt.xlabel('Date')
plt.ylabel('Weekly Sales')
plt.show()


prophet_model.plot_components(forecast)
plt.show()
```





Metrics

In [51]:
```
pred_arima = forecast_12_to_24_arima.values
pred_sarima = forecast_12_to_24_sarima.values
pred_prophet = forecast_12_to_24_prophet['Forecast'].values

actual = dataset1['Weekly_Sales_scaled'].iloc[-12:].astype(float)


pred_arima = pred_arima['yhat'] if 'yhat' in pred_arima else pred_arima.squeeze()
pred_sarima = pred_sarima['yhat'] if 'yhat' in pred_sarima else pred_sarima.squeeze
pred_prophet = pred_prophet['yhat'] if 'yhat' in pred_prophet else pred_prophet.squ
```

In [52]:
```
pred_prophet_numeric = pred_prophet
```

In [53]:
```
actual_12 = actual[-12:]
pred_prophet_12 = pred_prophet_numeric[:12]
```

In [54]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
def evaluate_model(actual, predicted):
    mae = mean_absolute_error(actual, predicted)
    mse = mean_squared_error(actual, predicted)
    rmse = np.sqrt(mse)
    return [mae, mse, rmse]
metrics = {
    'ARIMA': evaluate_model(actual_12, pred_arima),
    'SARIMA': evaluate_model(actual_12, pred_sarima),
    'Prophet': evaluate_model(actual_12, pred_prophet_12)
}
```

```
results = pd.DataFrame(metrics, index=['MAE', 'MSE', 'RMSE'])
print(results)
```

```
            ARIMA        SARIMA       Prophet
MAE   8.181165e+05  8.267784e+05  1.053438e+06
MSE   6.694483e+11  6.849553e+11  1.111303e+12
RMSE  8.181982e+05  8.276203e+05  1.054183e+06
```

Increase in sales

In [55]:
```
forecast_12_to_24_prophet_numeric = forecast_12_to_24_prophet['Forecast']
```

In [56]:
```
actual_last_12 = dataset1['Weekly_Sales'].iloc[-12:]
avg_actual = actual_last_12.mean()
```

In [57]:
```
increase_arima = ((forecast_12_to_24_arima.mean() - avg_actual) / avg_actual) * 100
increase_sarima = ((forecast_12_to_24_sarima.mean() - avg_actual) / avg_actual) * 1
increase_prophet = ((forecast_12_to_24_prophet_numeric.mean() - avg_actual) / avg_a


print(f"Expected % increase in weekly sales (weeks 12-24):")
print(f"ARIMA: {increase_arima:.2f}%")
print(f"SARIMA: {increase_sarima:.2f}%")
print(f"Prophet: {increase_prophet:.2f}%")
```

```
Expected % increase in weekly sales (weeks 12-24):
ARIMA: 12.08%
SARIMA: 13.27%
Prophet: 44.32%
```

In [64]:
```
forecast.to_csv('C:/Users/vanka/OneDrive/Documents/My_projetcs/project_3/prophet_fo
```

In [ ]:
```

```